Logica

0: Logica per l'Informatica

Claudio Sacerdoti Coen

<sacerdot@cs.unibo.it>

Universitá di Bologna

25/09/2018

Outline

Recapiti

- Prof. Claudio Sacerdoti Coen
  via Malaguti 1 (scala D)
  051 2094973
  claudio.sacerdoticoen@unibo.it
  `http://www.cs.unibo.it/~sacerdot`

- Pagina Web del corso:
  `http://www.cs.unibo.it/~sacerdot/logica`
- Orario di ricevimento (USATELO!!!):
  Giovedì 14:30

Cosa faccio?

Docente di

- Logica per l'Informatica
  (triennale informatica, mutuato filosofia)
- Fondamenti Logici dell'Informatica
  (magistrale informatica)
- Paradigmi emergenti di programmazione
  (magistrale informatica)

Cosa faccio?

Attività di ricerca:

- Teoria dei tipi e dimostrazione assistita
  (software per verificare la correttezza di dimostrazioni, per esempio di correttezza del codice di programmi)
- Linguaggi di programmazione logici, di ordine superiori e a vincoli
- Mathematical Knowledge Management
  (software per la gestione e sfruttamento di conoscenze matematiche)

Ma lei programma?

Paradigmi di programmazione preferito:

- funzionale (OCaml, Haskell, Lisp, ...)
- logico (λProlog, ...)

Linguaggio di programmazione preferito:

- OCaml ($\approx$ 130,000 righe di codice)
- λProlog ($\approx$ 15,000 righe di codice)

Sistema operativo preferito:

- Linux, what else?

Outline

Cos'è la logica?

Una disciplina molto ampia con contributi e applicazioni

1. In filosofia
2. In matematica
3. In informatica

Alcune domande chiave:

- cos'è la correttezza di un ragionamento?
- quali ragionamenti sono corretti?
- ci sono fatti non deducibili tramite un ragionamento?

L'informatica come studio di ciò che si può calcolare è figlia della logica ($\approx$ 1930).

"Il computer sta all'informatica
come il microscopio sta alla biologia."

La seguente dimostrazione è corretta?

Definition 1. Let H be a subgroup of a group G. A *left coset* of H in G is a subset of G that is of the form xH, where $x \in G$ and $xH = \{xh : h \in H\}$. Similarly a *right coset* of H in G is a subset of G that is of the form Hx, where $Hx = \{hx : h \in H\}$

Note that a subgroup H of a group G is itself a left coset of H in G.

Lemma 1. *Let H be a subgroup of a group G, and let x and y be elements of G. Suppose that $xH \cap yH$ is non-empty. Then $xH = yH$.*

Proof. Let z be some element of $xH \cap yH$. Then $z = xa$ for some $a \in H$, and $z = yb$ for some $b \in H$. If h is any element of H then $ah \in H$ and $a^{-1}h \in H$, since H is a subgroup of G. But $zh = x(ah)$ and $xh = z(a^{-1}h)$ for all $h \in H$. Therefore $zH \subset xH$ and $xH \subset zH$, and thus $xH = zH$. Similarly $yH = zH$, and thus $xH = yH$, as required. □

Lemma 2. *Let H be a finite subgroup of a group G. Then each left coset of H in G has the same number of elements as H.*

Proof. Let $H = \{h_1, h_2, \ldots, h_m\}$, where $h_1, h_2, \ldots, h_m$ are distinct, and let x be an element of G. Then the left coset xH consists of the elements xh_j for

Cos'è una dimostrazione?

Dimostrazione = sequenza di frasi che convincono il lettore che l'enunciato (tesi) valga.

Quando sono i singoli passi corretti?

- Individueremo linguaggi artificiali per scrivere i passi
  Un computer potrà dire se sono corretti
- Corretto = se valgono le premesse allora vale la conclusione

Ma cosa vuol dire valere?

La logica studia la verità?

A vale quando *A* è vera?

- In un libro fantasy possono esserci ragionamenti sugli unicorni rosa volanti.
- Il ragionamento può essere corretto o sbagliato.
- Ma di sicuro non ci parla di verità

Ma allora di cosa ci parla la logica?

Correttezza e completezza

Supponiamo di aver capito cosa significa “vale”.

1. Correttezza: tutto ciò che è dimostrabile vale?
   Ovvero, le nostre regole sono corrette?
2. Completezza delle regole: tutto ciò che vale è dimostrabile?
   Ovvero, abbiamo messo abbastanza regole?
   Se risposta negativa: cosa si può dimostrare e cosa no?
3. Completezza delle ipotesi: aggiungendo ipotesi posso dimostrare tutto ciò che vale in una determinata situazione?
   Risposta negativa: cosa non potremo mai sapere e perchè?

Logica e automazione

Può un programma dirci se una dimostrazione è corretta?

Può un programma fare dimostrazioni automaticamente?

Se no, perchè?

Se sì, con quale complessità computazionale? (di quanto spazio e tempo ha bisogno)?

Una, dieci, cento logiche?

Il senso che diamo a "valere" determina le regole della logica e delle dimostrazioni in tale logica

Tante interpretazioni di valere $\Rightarrow$ tante logiche

1. verità $\Rightarrow$ logica classica
2. evidenza/programmabilità $\Rightarrow$ logica intuizionista
3. accade $\Rightarrow$ logica temporale
4. conoscenza $\Rightarrow$ logica epistemica
5. possesso $\Rightarrow$ logica lineare
6. ...

Esempio: se X è vero/è noto/accade/è ritenuto vero/è posseduto e quando X lo è lo è anche Y, allora Y lo è?

Logica, la materia circolare

Fisica, astronomia, biologia, . . . sono basati sulla matematica e la logica.

Matematica e informatica sono basati sulla logica.

Studieremo la logica usando strumenti matematici e informatici basati su un'altra logica
che chiameremo **meta-logica** (= logica che parla di una logica).

Solo alla fine del corso comprenderete gli strumenti già usati fin dall'inizio!

Outline

Due materie, una fondazione comune

I matematici **Gli informatici**

risolvono **problemi** tramite

dimostrazioni **codice**

1. decomponendo il problema in problemi più semplici
2. risolvendo tali problemi con la stessa procedura
3. ottenendo la soluzione al problema di partenza
   componendo le soluzioni dei nuovi problemi
   (la ricomposizione è anch'essa un nuovo problema!)

La **logica** garantisce la correttezza del procedimento.

Esempio di decomposizione (1/3)

Una sequenza di numeri o è vuota `[]` o è un numero *n* seguito da una sequenza `L` (`n::L`).

Problema: scrivere una funzione `f(L)` che trovi una sotto-sequenza crescente in `L` di lunghezza massimale.

```
f([]) = []
f(n::L) =
 if length(initial(n,L)) > length(f(L)) then
   initial(n,L)
 else f(L)
```

Sotto-problemi:

1. scrivere una funzione `length(L)` che calcoli la lunghezza di `L`
2. scrivere una funzione `initial(n,L)` che restituisca la sequenza crescente iniziale di $n :: L$

Esempio di decomposizione (2/3)

Sotto-problema: scrivere una funzione `length(L)` che calcoli la lunghezza di `L`

```
length([]) = 0
length(n::L) = 1 + length(L)
```

Esempio di decomposizione (3/3)

Sotto-problema: scrivere una funzione `initial(n,L)` che restituisca la sequenza crescente iniziale di $n :: L$.

```
initial(n,[]) = n::[]
initial(n,m::L) =
  if n < m then n::initial(m,L)
  else n::[]
```

Due materie, problematiche comuni

Come faccio a

1. creare linguaggi artificiali di prova/programmazione?
2. decomporre la mia prova/programma in componenti riusabili?
3. creare librerie riusabili di teoremi/programmi?
4. astrarre le mie prove/programmi in modo che siano applicabili in contesti diversi?
5. riadattare una soluzione a un contesto diverso?
6. analizzare soluzioni a problemi distinti e fattorizzare le parti comuni?

Due materie, due enfasi differenti

I matematici sono interessati a **l'esistenza** delle soluzioni di un problema

Gli informatici sono interessati a il modo di **calcolare** le soluzioni di un problema

Due materie, due sensibilità differenti

Dr. Math: c'è sicuramente un valore x per la quale $f(x) = 0$
Mr. Inf: interessante, qual'è?
Dr. Math: non ne ho idea
Mr. Inf: ...
Mr. Inf: e come fai a esserne sicuro?
Dr. Math: semplice, ragioniamo per assurdo. Supponi che non ci sia. Poi scegli un y dall'insieme infinito dei numeri tali che ... QED
Mr. Inf: non ho ancora capito come calcolarlo però
Boring Logician: ovviamente, si dimostra che è impossibile scrivere un programma che lo calcoli!

Due materie, due pratiche differenti

Mr. Inf: guarda, questo programma calcola tutti i modi di colorare una cartina geografica usando solo 4 colori! fanno 1000 euro
Dr. Math: interessante, come fai a esserne sicuro?
Mr. Inf: beh, ovvio, ci ho pensato quando ho scritto il codice e passa tutti i test
Dr. Math: . . .
Dr. Math: guarda, su questa cartina sbaglia
Mr. Inf: ah, beh, sì, aspetta che lo fisso. Ecco ora funziona
Dr. Math: e perchè mai ora dovrebbe?
Boring Logician: beh, se proprio ci tenete esplicito io la prova di correttezza e vi trovo tutti i bug, per solo 10000 euro e 4 anni di lavoro!

Informatica $\subseteq$ matematica

Un programma la cui logica sia completamente esplicitata è una dimostrazione matematica

I bug logici sono causati da errori logici nella decomposizione/ricomposizione che sarebbero evitati esplicitando la dimostrrazione

Imparare a ragionare logicamente formulando nella vostra mente (parti del)le prove semplifica la scrittura di codice e riduce gli errori

Informatica $\subsetneq$ matematica

La matematica studia

1. oggetti infiniti (non rappresentabili in memoria, non manipolabili algoritmicamente)
2. funzioni e trasformazioni che si dimostrano impossibili da calcolare
3. funzioni e trasformazioni che possono essere calcolate, ma solo avendo a disposizione quantità smoderate di risorse (spazio, tempo, soldi, . . .)

Altre motivazioni

La logica trova ulteriori applicazioni in informatica:

- architettura, circuiti logici
- basi di dati
- intelligenza artificiale
- studio e sviluppo di linguaggi di programmazione
- verifica automatica o manuale di proprietà del codice
- linguaggi di programmazione logici
- analisi sull'uso delle risorse

La correttezza del codice

Quando un software è corretto?

- specifica di un software =
  descrizione di cosa il software deve fare
- semantica di un linguaggio di programmazione =
  descrizione di come si eseguono i programma
- descrizione = formula logica!
- software corretto =
  il codice fa quello che dice la specifica =
  una formula logica implica l'altra
- software corretto se e solamente se c'è una DIMOSTRAZIONE CORRETTA che quello che il codice fa è quello che dovrebbe fare

Perchè circola software non corretto?

Programmare è l'attività più complessa mai ideata dall'uomo

Che succede se viene giù un ponte?
E se un software va in crash?

- *Programmare* $\approx$ *dimostrare*
  vedi corso magistrale Fondamenti Logici dell'Informatica
  Cosa c'è di più complesso?
- *Certificazione automatica solo parziale*
  I computer privi di intuizioni non possono trovare dimostrazioni

Perchè non si fanno dimostrazioni manuali/assistite allora?

Perchè circola software non corretto?

La certificazione manuale costa!

- dati Intel: $\approx$ 10x mesi uomo per dimostrazione interattiva assistita rispetto a implementazione + testing
- usata solo per codice critico

Codice critico = tale per cui un bug mette a rischio ingenti quantità di soldi (non vite umane...)

Esempi: controllo aereo, centrali nucleari, smartcard, software medicale, aritmetica floating point dei microprocessori, ...

Outline

Orari delle lezioni

- Lezioni frontali:
  - gio + ven, 9:00(+15)–11:00(-15?), aula M1
- Laboratorio:
  - non tutte le settimane; si inizia il ??/?? in lab Ercolani
  - AL: 14:00-16:00(-15), MZ: 16:00–18:00, alternati
  - gruppi da 2 persone
  - uso del dimostratore interattivo di teoremi Matita, versione 0.5.9 in ambiente Linux
  - oppure: esercitazioni cartacee
  - possibilità di arrotondare il punteggio terminando “a casa”

Modalità di esame

Prova scritta ove vengono richieste:

- definizioni, enunciati, dimostrazioni
- esercizi di dimostrazione e costruzione di modelli

Voto finale =
media pesata prova scritta +
punteggio conseguito in laboratorio

Orale integrativo

- obbligatorio per chi non frequenta il laboratorio
- possibile per chi consegue 16-17 punti allo scritto
- possibile per chi vuole migliorare il voto

Materiale didattico

Slides: sul sito http://cs.unibo.it/ sacerdot/logica/
messe a disposizione durante il corso

Libro di testo: “Logica a Informatica”, Asperti, Ciabattoni
le lezioni non seguiranno l’ordine del libro

Outline

Decalogo dello studente furbo

1. evitare accuratamente di prendere appunti a lezione!

Il docente è pagato per dire cose inutili e non interessanti.
Si rischia inoltre di farsi venire dubbi che rallentano lo studio.

Decalogo dello studente furbo

2. non ripassare mai durante l'anno!

Riguardare i lucidi prima della lezione dopo è una perdita di tempo, anche se ogni lezione riusa le definizioni e le notazioni delle precedenti, e non averli memorizzati significa non capire nulla.

Decalogo dello studente furbo

3. non andare mai a ricevimento durante l'anno!

Andare a ricevimento serve per rimettersi in pari su quello che non si è capito. Basta fregarsene di capire durante l'anno per evitare il ricevimento.

Decalogo dello studente furbo

4. non fate mai domande in aula!

Se farete domande il docente potrebbe capire che non avete capito o che ha spiegato male, per poi punirvi rispiegando. Peggio ancora se doveste imparare qualcosa che poi non vi serva all'esame. Per scrupolo, evitate anche accuratamente di rispondere quando il docente fa domande o sondaggi.

Decalogo dello studente furbo

5. non chiedetevi mai il perchè!

L'importante è imparare tutto a memoria senza sapere a cosa servono le definizioni e perchè i teoremi sono importanti. In ogni caso dimenticherete tutte le dimostrazioni dopo il corso (e le potete ritrovare sul Web), quindi perchè mai capire il senso dei concetti che vi servirebbe in futuro per sapere come ritrovarli e come applicarli?

Decalogo dello studente furbo

6. imparate a fare gli esercizi, non la teoria!

Agli esami vi chiedono di fare esercizi meccanici, no? Tanto per fare un esercizio meccanico non c'è bisogno di sapere cosa si sta facendo. La teoria è per quelli stupidi che poi cercano di risolvere problemi nuovi. E chi se ne frega se poi verrete rimpiazzati in azienda da un robot che fa le cose meccaniche più velocemente e per giunta correttamente.

Decalogo dello studente furbo

7. iniziate a studiare sette giorni prima dell'esame!

Perchè andare con calma e rischiare di assimilare le lezioni bruciando neuroni utili quando potete fare una tirata intensiva solo per passare l'esame, con la certezza di dimenticare tutto subito dopo?

Decalogo dello studente furbo

8. provate e riprovate gli esami senza studiare!

Studiate in Italia, l'unico posto meraviglioso che offre ben sei appelli l'anno per materia. Se ci sono, vuol dire che dovete usarli tutti, no? Continuate a provare l'esame senza studiare fra un appello e l'altro, tanto prima o poi passerete. E se non dovesse succedere, tutti i docenti a un certo punto cedono per sfinimento e un 18 ve lo danno! Ok, magari un giovane laureato trentenne con la media del 18 non è il massimo, ma perchè strafare?

Decalogo dello studente furbo

9. copiate!

Imparate a copiare prima che potete: magari sarà l'unica abilità che conseguirete all'università, ma vi tornerà utile di sicuro. Fatelo soprattutto quando siete così persi da non capire nemmeno che state copiando da uno che sta sparando le risposte a caso. E poi si sà, i docenti sono sempre di buon umore quando correggono per la trentesima volta la stessa soluzione sbagliata. Per non parlare dei datori di lavoro che assumono due informatici per fare il lavoro di mezzo.

Decalogo dello studente furbo

10. date prima gli esami facili!

Più esami facili darete per primi, più vi sembrerà di fare progressi in fretta. Non vi preoccupate di quando arriverete ad avere solo gli esami super-difficili e lunghi da dare.